§ 1.136(a), and any fees required therefor (including fees for net addition of claims) are hereby authorized to be charged to our Deposit Account No. 19-0036.

Amendments

In the Specification:

Please substitute the paragraph beginning on page 1, line 7, with the following paragraph:

A1

U.S. Patent Application No. 09/363,637; inventors Ying-wai Ho, Michael Schulte and John Kelley; and entitled "System and Method for Improving the Accuracy of Reciprocal and Reciprocal Square Root Operations Performed by a Floating-Point Unit;"

Please substitute the paragraph beginning on page 1, line 12, with the following paragraph:

A2

U.S. Patent Application Serial No. 09/364,514; inventors John Kelley and Ying-wai Ho; and entitled "Floating-Point Processor With Improved Intermediate Result Handling;"

Please substitute the paragraph beginning on page 1, line 16, with the following paragraph:

A3

U.S. Patent Application Serial No. 09/364,787; inventors Radhika Thekkath, Michael Uhler, Ying-wai Ho, and Chandlee Harrell; and entitled "Processor Having an Arithmetic Extension of an Instruction Set Architecture;"

Please substitute the paragraph beginning on page 1, line 21, with the following paragraph:

A4

U.S. Patent Application Serial No. 09/364,789; inventors Radhika Thekkath, Michael Uhler, Ying-wai Ho, and Chandlee Harrell; and entitled "Processor Having a Conditional Branch Extension of an Instruction Set Architecture;"

Please substitute the paragraph beginning on page 2, line 2, with the following paragraph:

A5

U.S. Patent Application Serial No. 09/364,512; inventors Ying-wai Ho, John Kelley and James Jiang; and entitled "Processor With Improved Accuracy For Multiply-Add Operations;" and

Please substitute the paragraph beginning on page 2, line 6, with the following paragraph:

A6

U.S. Patent Application Serial No. 09/363,638; inventors James Jiang, Ying-wai Ho and John Kelley; and entitled "Method and Apparatus for Predicting Floating-Point Exceptions."

Please substitute the paragraph beginning on page 29, line 1, with the following paragraph:

A7

Floating point adder 284 is a floating point mantissa adder which implements single precision, double precision, and paired-single floating point add instructions (e.g., ADDR of Table 1) and subtract instructions, as well as the add/subtract portions of compound instructions such as MADD (i.e., floating point multiply add, described below). Floating point adder 284 accepts two operands, an intermediate result from floating point multiplier 283 and a mantissa staged in floating point pipe file 282. To increase performance, a floating-point magnitude addition/subtraction operation is computed by either a prescale adder (PSA) 583 or a massive cancellation adder (MCA) 584 (Figure 2D). PSA 583

performs all magnitude additions and often performs magnitude subtractions if the difference in operand exponents is greater than two (2), thereby avoiding a large normalization shift after the operation. MCA 584 often performs magnitude subtractions if the difference in operand exponents is less than or equal to two (2), thereby avoiding a large alignment shift before the operation. Thus, the final correct result is selected from either PSA 583 or MCA 584 based upon, among other things, the exponential difference of the operands. The result is then returned to floating point pipe file 282. Selection criteria for PSA 583 and MCA 584 are further described in the above-referenced U.S. Patent Application Serial No. 09/364,512.

A1
cont.

Please substitute the paragraph beginning on page 32, line 6, with the following paragraph:

Referring to Figure 2D, data start from floating point register file 281 passing from register 502 (32-entry, 64-bit register file with 4 read ports and 2 write ports) to unpack/bypass logic 508 in pipe file 282. (Data may also flow directly into logic 508 from load bus 291 and register file 507.) This logic unpacks an operand into an "internal format," discussed in previously-identified copending application nos. 09/363,638 and 09/363,637. This logic may also perform bypass operations for operands that do not require any arithmetic operation (i.e., circulate operands back to file 281). Where arithmetic operation is required, data then flow to multiplier 283 and exponent 286.

A6

Please substitute the paragraph beginning on page 33, line 16, with the following paragraph:

A9

Further discussion of FPU 270 and alternative embodiments are provided in the previously identified copending application nos. 09/364,514, 09/364,512, 09/363,638, and 09/363,637.

Please substitute the paragraph beginning on page 36, line 18, with the following paragraph:

A10

An alternative embodiment of FCSR 410 is provided in previously-identified copending application no. 09/364,512. In this embodiment, an additional control bit "FO" (Madd-flush-override bit) is provided to the FCSR. The combination of bits FS and FO enable an FPU (such as FPU 270) to selectively operate in up to three different modes; i.e., IEEE-compliant, Flush-to-zero and Madd-flush-override.

Please substitute the paragraph beginning on page 52, line 18, with the following paragraph:

A11

Equation (1) results in a term "(2 - bx_i)" which is frequently close to 1.0 (such as 1.0000...nnnn..., where nnnn is the correction adjustment and the number of interest). This format can result in a loss of precision. In contrast, in equation (2), the term "(1 - bx_i)" is first determined (using RECIP2 604). By subtracting 1.0 in RECIP2 604, the resulting number is typically very small and can be normalized (as n.nnn...) to achieve greater precision. Therefore, subsequent calculations are more precise and denormalization may be avoided. This technique is further discussed in copending application no. 09/363,637.

Please substitute the paragraph beginning on page 64, line 1, with the following paragraph:

A12

Equation (3) has a term x_i^2. If b is a very large number, x_i^2 can initially create a denormalized number. In contrast, in equation (4) the term "$(1-bx_i*x_i)/2$" is determined

using RSQRT2 606 in a sequence of instructions (as set out below) where b is first multiplied with x_i. This order of operation can pull the quantity "$b*x_i$" back to a sufficiently normal range so that subsequent multiplication with x_i may not create denormalization. This technique is further discussed in copending application no. 09/363,637. This operation is computed in the pipelines of floating point multiplier 283 and floating point adder 284 of FPU 270 (Figure 2C).

A1 cont.